Reachability Questions? Developers Love This One Weird Tool

James Yoo
MSc Thesis Presentation
October 17, 2022
Developers Developers Developers

What do they do, really?
What do developers do? (really)

What do developers do? (really)

- Editing code: 20%
What do developers do? (really)

- 28% Other
- 20% Editing code

What do developers do? (really)

28% Other

52% Program comprehension activities

20% Editing code

What Questions Do Developers Ask?

Survey of 460 software engineers at Microsoft

• Developers ask *reachability questions*
  
  • Asked them more than *9 times a day*
  
  • 82% of them reported them as being somewhat hard to answer

Field study of 17 developers

• Reachability questions accounted for *9 of the 10 longest activities*
Developers Ask Reachability Questions
LaToza and Myers’ Framework

find SC in TR

• Looking through traces to find points of interest
• “Where is this method generating an error?”
• “How is data being mutated between these two method calls?”
LaToza and Myers’ Framework

find SC in TR

- Looking through traces to find points of interest
- “Where is this method generating an error?”
- “How is data being mutated between these two method calls?”

compare(TRa, TRb) : TRcommon, TR1, TR2

- Given two traces, find what is common, and what is unique to each trace
- “What’s the difference between the test and production version of this app?”
Direct Tool Support for Reachability Questions

REACHER [1]

• Focuses on control-flow graph exploration
• Invoked via menu

Get Me Here [2]

• Reframes reachability questions to SMT problems
• Invoked via menu
• No user study on usability

Reachability Questions in Practice

RQ1 What are the reachability questions frequently encountered by developers in practice?
Reachability Questions in Practice

RQ1 What are the reachability questions frequently encountered by developers in practice?

Survey of 72 practicing software developers

- 9 reachability questions from the literature, past experiences
- Distributed via Twitter
This figure represents a scenario where some data is created (the `cart` object), and passed along as an argument to some methods. It may not be immediately clear how this data is being modified (e.g., which fields of the `cart` object are being updated or changed).

```java
public ShoppingCart provision() throws CartValidationException {
    logger.info("Provisioning new cart");
    ShoppingCart cart = CartBuilder.build();
    if (this.hasValidState(cart)) {
        this.register(cart);
        return cart;
    }
    throw new CartValidationException("Provisioning a new cart failed");
}
```

Are you interested in how `cart` could be modified in these method calls?

---

How often do you ask yourself this question:
Given some data (in this case, `cart`), which parts of it are modified downstream?

<table>
<thead>
<tr>
<th>I never ask myself this question</th>
<th>I often ask myself this question</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>
Reachability Questions in Practice

RQ1  What are the reachability questions frequently encountered by developers in practice?

Survey of 72 practicing software developers

- 9 reachability questions from the literature, past experiences
- Distributed via Twitter
Reachability Questions in Practice

RQ1 What are the reachability questions frequently encountered by developers in practice?

Survey of 72 practicing software developers

- 9 reachability questions from the literature, past experiences
- Distributed via Twitter

Developers frequently encounter reachability questions that are related to the data-flow of a program during their work
private ValidationInfo getValidationInfo() {
    FindModel findModel = new FindModel();
    this.directory = findModel.getDirectoryName();
}
@Nullable

public ValidationInfo validate(@NonNull FindModel model) {
    VirtualFile directory = FindInProjectUtil.getDirectory(model);
    if (directory == null) {
        return new ValidationInfo(FindBundle.message( key: "find.directory.not.found.error"), myDirectoryComboBox);
    }
    return null;
}
@Nullable

public ValidationInfo validate(@NonNull FindModel model) {
    VirtualFile directory = FindInProjectUtil.getDirectory(model);
    if (directory == null) {
        return new ValidationInfo(FindBundle.message("find.directory"), myDirectoryComboBox);
    }
    return null;
}
@Nullable
public ValidationInfo validate(@NonNull FindModel model) {
    VirtualFile directory = FindInProject Util.getActiveDirectory();
    if (directory == null) {
        return new ValidationInfo(FindBundle.message("find.empty.search.text.error"), mySearchOptions);
    }
    return null;
}

@Override
@NonNull
public String getFileTypeMask() {
    String mask = null;
    if (header.cbFileFilter != null && header.cbFileFilter.isSelected()) {
        mask = (String) header.fileMaskField.getSelectedItem();
    }
    return mask;
}

@Nullable("null means OK")
private ValidationInfo getValidationInfo() {
    FindModel findModel = new FindModel();
    this.directory = findModel.getActiveDirectory();
    ValidationInfo scopeValidationInfo = myScopeUI.validate(findModel, mySelectedScope);
    if (scopeValidationInfo != null) {
        return scopeValidationInfo;
    }
    if (!myHelper.canSearchThisString()) {
        return new ValidationInfo(FindBundle.message("find.empty.search.text.error"), mySearchOptions);
    }
}

FindPopupPanel.java
The End
Developer-centric needs

*What’s missing?*
Developer-centric needs

What’s missing?

Tool development

Maybe this?
Developer-centric needs
*What’s missing?*

Tool development
*Maybe this?*

Evaluation
*Did we build the right thing?*
Evaluating ReachHover

**RQ2** Are developers able to answer data-flow reachability questions more correctly with ReachHover?

**RQ3** Does ReachHover make it easier to answer data-flow reachability questions?
Evaluating ReachHover

RQ2  Are developers able to answer data-flow reachability questions more correctly with ReachHover?

RQ3  Does ReachHover make it easier to answer data-flow reachability questions?

Controlled experiment with 20 software developers

- Two tasks posing data-flow reachability questions
- Remote study with on-device logging
What are we measuring?

Correctness

• Scoring answers to reachability questions

\[ S = \text{ANS}_{\text{correct}} - \text{ANS}_{\text{incorrect}} \]

Visual Momentum

• Editor cursor clicks and jumps, proxy for context switching

Developer Experience

• Open coding on answers to two experiential questions
Task Example

How is a value created?

1. In how many locations does a call to the BookmarkInfo constructor exist?

2. Provide the names of the methods where the value of \textbf{bookmark} is used.

3. Provide the names of the methods where \textbf{bookmark} is assigned a value.
Average Correctness Score
Backward, inter-file reachability question

- ReachHover: 4.1
- Built-in Data-flow: 0.8
Average Correctness Score
Forward, intra-file reachability question

- ReachHover: 4.6
- Built-in Data-flow: 4.1
Visual Momentum

- Fewer cursor jump actions in the main editor on average for each task
- Statistically significant difference for the forward intra-file task
  \((\text{Mann-Whitney } U = 6, n = 10, p = 0.001, \alpha = 0.05)\)

Bar chart showing:
- Backward inter-file: ReachHover 28, IntelliJ 76.1
- Forward intra-file: ReachHover 49.5, IntelliJ 21.1
Developer Experience

EQ1 Did you find any difference using ReachHover or IntelliJ for the tasks in this study?
Developer Experience

**EQ1**  Did you find any difference using ReachHover or IntelliJ for the tasks in this study?

19 total responses

- **13** (65%) reported a positive experience with ReachHover
  - Easier and simpler to use (5)
  - Helps maintain context (5)
  - More usable (3)
- Some reported no difference (4), while others (2) preferred the built-in IDE features
“ReachHover felt less cluttered and made the task easier” – P18

“ReachHover was easier to call as it didn’t require navigating menus” – P9

“I find having ReachHover’s separate floating windows with a view into where the related line of code is, is more usable than IntelliJ’s data-flow analysis feature” – P10
“ReachHover felt less cluttered and made the task easier” – P18

“I find having ReachHover’s separate floating windows with a view into where the related line of code is, is more usable than IntelliJ’s data-flow analysis feature” – P10

“ReachHover was easier to call as it didn’t require navigating menus” – P9
“I preferred that I could look at the matches in the same modal window without affecting the open file” – P19

“I felt like I did not lose the context using ReachHover compared to IntelliJ. I felt they showed me the same content, but ReachHover was easier to use (maybe because [sic] I didn’t [sic] lose the context)” – P15
“I preferred that I could look at the matches in the same modal window without affecting the open file” – P19

“I felt like I did not lose the context using ReachHover compared to IntelliJ. I felt they showed me the same content, but ReachHover was easier to use (maybe because I didn’t lose the context)” – P15
Evaluating ReachHover

RQ2  Are developers able to answer data-flow reachability questions more correctly with ReachHover?

RQ3  Does ReachHover make it easier to answer data-flow reachability questions?
Evaluating ReachHover

ReachHover enabled participants to answer an inter-file data-related reachability question more accurately and an intra-file question as accurately, as built-in data-analysis support in an IDE.

RQ3 Does ReachHover make it easier to answer data-flow reachability questions?
Evaluating ReachHover

ReachHover enabled participants to answer an inter-file data-related reachability question more accurately and an intra-file question as accurately, as built-in data-analysis support in an IDE.

We observed a lower average of context-shifting main editor interactions when ReachHover was used compared to built-in IDE support. The majority of our participants found ReachHover to be easier or simpler to use, more usable, and helpful in preserving their context during tasks.
Discussion and Future Work
Discussion and Future Work

Asking and Answering Additional Reachability Questions

- Investigate *compare*-type questions
- Run-time information for dynamic reachability questions
- Use test execution data, production logs, dynamic slicing